FINITE IMPULSE RESPONSE FILTER METHOD AND APPARATUS

BACKGROUND OF THE INVENTION

1. Field of Invention

[0001] This invention is directed to a finite response filter (FIR) method and apparatus.

2. Related Art

[0002] FIR filters are used in many fields such as document reproduction or printing of documents, for example. Almost all printed matter uses halftone screens. These halftone screens are traditionally optimized for the printing device, and may cause considerable halftone interference, such as visible large-area beating and visible Moire patterns, if not properly removed from the original scanned image. Image data generated by scanning printed matter is often filtered to remove unwanted artifacts, such as halftone screens. The suppression of halftones is especially important for color documents, since these are typically printed with four or more color separations containing slightly different screens at different angles and or frequencies, and these may interact with each other to cause undesirable spatial artifacts.

SUMMARY OF THE DISCLOSURE

[0003] A system and method for filtering image data is provided that takes advantage of FIR filter properties. For example, a two-dimensional triangular FIR filter having N x N coefficients can be applied to image data so that consecutive output pixels are generated by processing N x N blocks of image data pixels shifted a constant number of pixels from each other. Coefficients of such a filter may be divided into four quad portions, each quad portion corresponding to a quadrant of the N x N coefficients. Each of the four quad portions of coefficients may be used to separately generate a partial filter result for a single N/2 x N/2 block of image data. The partial filtered results may be stored in a memory and later retrieved to generate complete filter outputs. Such a filter process may be applied to de-screening image data in document processing applications, for example.

BRIEF DESCRIPTION OF DRAWINGS

[0004] The systems and methods of this invention are described in detail, with reference to the following figures, wherein:

[0005] Fig. 1 illustrates exemplary coefficients of a two-dimensional triangular filter with a span N of seven;

[0006] Fig. 2 illustrates an exemplary zero padded filter coefficients to convert odd Ns to an even number;

[0007] Fig. 3 illustrates dividing the padded fulter coefficients of Fig. 2 into four quad portions forming four quad filters;

[0021] Fig. 4 illustrates applying the quad filters to an N/2 x N/2 block of image data pixels;

[0021] Fig. 5 shows an exemplary process of quad filtering the image data;

[0008] Figs. 6 and 7 show an exemplary relationship between N/2 blocks and filter outputs;

[0009] Fig. 8 shows an exemplary filter processor;

[0010] Figs. 9-11 show exemplary flow charts of the filter process.

DETAILED DESCRIPTION

[0011] As is well known, two-dimensional FIR filters may be used to filter image data. When applying such a filter that has N x N coefficients, image data pixels corresponding to an N x N block is multiplied by appropriate coefficients and the products summed to generate a single filter output pixel. When the coefficients are symmetrical about a central point, the filter coefficients may be divided into quadrants and each of the quadrants (a quad filter) may be applied to an N/2 x N/2 block of the image data separately to generate partial filtered results. These partial filtered results may be combined to form complete filter outputs without again accessing all corresponding pixels of the N x N block of the image data.

[0012] Fig. 1 shows coefficients 100 of an exemplary two-dimensional triangular FIR filter. While the coefficients 100 corresponds to an odd N that equals 7, N can be any number, even or odd, such as 30 or 31, for example. When N is even and the coefficients are symmetrical about the center, the coefficients may be divided into 4 quad portions of N/2 x N/2 blocks, and each of the quad portions may be separately applied as a quad filter to N/2 x N/2 blocks of image data. However, if N is odd, as shown in Fig. 1,

then a row of zeros (0) and a column of zeros (0) may be added (zero padding), as shown in Fig. 2, and the N+1 x N+1 block of coefficients 102 may be divided into 4 (N+1)/2 x (N+1)/2 quad portions 110, 120, 130 and 140 as shown in Fig. 3. While fig. 2 shows the row and column of zeros padded at the top and left sides of the N x N coefficients, right and bottom sides may zero padded instead of the top and left sides.

[0013] Fig. 4 shows a single filter process that filters N/2 x N/2 blocks of pixels 160 of the image data generating partial filtered results 210-240 using the 4 quad portions 110-140 as coefficients of 4 quad filters. The four quad filters may be combined into a single partial filter 150, as an example. The partial filter 150 processes on each of the N/2 x N/2 blocks of pixels 160 of the image data to generate the 4 partial filtered results 210-240 (which is also collectively referenced as partial filtered results 180). These partial filtered results 180 may be stored in a memory for later retrieval and combined to generate complete filter results. Different ones of the partial filtered results 210-240, generated by the partial filter 150 from the same N/2 x N/2 block, are combined with partial filtered results 210-240 of other N/2 x N/2 blocks to generate complete filter outputs.

[0014] If the number of complete filter output pixels is required to match the number of pixels in the image data, then (N/2) -1 pixels along a perimeter of an image area corresponding to the image data may be replicated along edges of the image area; and every N/2 x N/2 block of the expanded image data may be processed to generate the complete filter output. However, the amount of information in the complete filter output is less than that in the original image data because information was filtered out by the filter process. Thus, fewer pixels are required to represent the contained information. Accordingly, a number of pixels in the complete filter output may be reduced or decimated by sub-sampling by a factor k^2:1, for example. The sub-sampling may be accomplished by selecting N/2 x N/2 blocks of pixels that are shifted from each other by k pixels. For example, if the image data was obtained by scanning a document, the N/2 x N/2 blocks may be shifted in a fast scan direction (from left to right) by k pixels increments. When the edge of the image area is reached, the left most N/2 x N/2 block shifted downward by k pixels may be selected to begin another pass in the fast scan direction.

[0015] Fig. 5 shows the partial filter 150 when applied to the image data when k = N/2. Image data 162 is shown divided into N/2 x N/2 blocks with indexes at the top and left sides for each N/2 increment. Using the notation (left index top index) to refer to a particular N/2 x N/2 block, (0 0) refers to the top left N/2 x N/2 block; (0 (I-1)) refers to the top right most N/2 x N/2 block; ((J-1) 0) refers to the left most bottom N/2 x N/2 block; and ((J-1) (I-1)) refers to the right most bottom N/2 x N/2 block.

[0016] The N/2 x N/2 blocks of the image data 162 are filtered by the partial filter 150 to generate partial filtered results 182 indexed by the N/2 x N/2 block indexes (j i). Thus, partial filtered results 210-240 of (0 0) corresponds to the (0 0) N/2 x N/2 block in the image data 162. The partial filter 150 may process the image data 162 along a fast scan direction (left to right) N/2 x N/2 block at a time until the right edge of the image data 162 is reached, and then begin again at the left most N/2 x N/2 block one N/2 x N/2 block down in the slow scan direction, for example. Thus, N/2 x N/2 blocks (0 0) to (0 (I-1)) blocks are processed by the partial filter 150 from left to right, then (1 0) to (1 (I-1)) are processed next to generate the partial filtered results 182.

[0017] Fig. 6 shows an exemplary diagram of the image data 162 of Fig. 5 overlaid with corresponding filtered output pixels (squares filled with dots). The top and left numbers index the output pixels. Thus, filtered output pixel (0 0) is generated by combining partial filtered results corresponding to N/2 x N/2 blocks (0 0), (0 1), (1 0) and (1 1) shown in Fig. 5. Fig. 7 shows the partial filtered results 210-240 that are combined for each of the filtered output pixels (0 0), (0 1), (0 2), (1 0), (1 1) and 1 2). Thus, the top left most filtered output pixel is generated by:

1) multiplying each of the coefficients 210 by pixel values of the (0 0) N/2 x N/2 block of the image data and summing the products;

2) multiplying each of the coefficients 220 by pixel values of the (0 1) N/2 x N/2 block of the image data and summing the products;

3) multiplying each of the coefficients 230 by pixel values of the (1 0) N/2 x N/2 block of the image data and summing the products;

4) multiplying each of the coefficients 240 by pixel values of the (1 1) N/2 x N/2 block of the image data and summing the products; 5) summing all the sums of the products; and optionally

6) normalizing by dividing by the sum of all the coefficients 110-140.

[0018] The division in the normalization process made be made efficient if the sum of all the N x N coefficients is a power of 2, i.e., 4, 16, 32, etc. If so, the normalization may be performed by one or more binary right shifts. Thus, if the sum of the coefficients are 4, 16 or 32, right shifts of 2, 4 or 5 are performed for normalization. Rounding may be achieved by adding to the sum of the sums of the products a value equal to half the sum of all the coefficients before right shifting for the divide. Since the right shifts performs the division by the sum of the coefficients, adding one half the sum of the coefficients. prior to the right shifts is effectively adding .5 to the division result.

[0019] If the sum of the coefficients is not a power of 2, then normalization may be approximated by multiplying the sum of the sums of the products by a fraction that has a denominator of a power of 2. In this case, normalization is achieved by one multiplication followed by one or more right shifts.

[0020] Fig. 8 shows an exemplary block diagram of a filter device 300 that filters the image data based on the quad filters described above. The filter device 300 includes a CPU 302, a partial filter result generator 304, a filter output generator 306, a memory 308 and an input/output port 310. All the components 302-310 are coupled together with a bus 312. The CPU 302 may include a control program that coordinates the functions of the other components 304-310 to perform the filter function.

[0021] While Fig. 8 shows a bus architecture, the filter device 300 may be implemented using any hardware structure that performs the needed functions. For example, the filter device may be implemented on a single application specific integrated circuit (ASIC), PLAs, and the like, or implemented in software as a program executing in a processor such as the CPU 302. The structure shown in Fig. 8 is used for ease of discussion.

[0022] Each of the quad filter coefficients 110-140 may be stored in the memory 308 together with their sums, a number of right shifts or a fraction expressed by a multiplier and a number of right shifts. The image data to be filtered may be received via the input/output port 310. For k = N/2, the image data may be received one N/2 x N/2 block at a time and discarded after the partial filtered results are generated. Thus, each N/2 x N/2 block of image data may be input and stored in the memory 308 until the partial filter result

generator 304 completes generating all the partial filtered results 210-240 corresponding to the N/2 x N/2 block of image data. If Input/output speeds are slow, a next N/2 x N/2 block may be stored in the memory 308 while the partial filtered results are being generated by the partial filter result generator 304.

[0023] The input/output port 310 may input the N/2 x N/2 blocks of image data in the fast scan and slow scan directions, as discussed above. Thus, referring to Fig. 5, N/2 x N/2 blocks of the image data 162 may be input from left to right and top to bottom until all the image data (or as much as needed) is processed by the filter device 300.

[0024] When a new N/2 x N/2 block of image data is received, the CPU 302 may command the Partial Filter Result Generator 304 to begin generating the partial filtered results. The partial filter result generator 304 may generate the 4 partial filtered results 210-240 by multiplying the quad coefficients 110-140 with the corresponding image data of the stored N/2 x N/2 block, summing the products, as discussed above. The partial filtered results 210-240 may be stored as shown in Fig. 5 in the indexed order. After all N/2 x N/2 blocks of image data are processed by the partial filter result generator 304, the partial filtered results 182 as shown in Fig. 5 is stored in the memory 308.

[0025] Not all the partial filtered results 182 may be stored in the memory 308 all at the same time because the filter output generator 306 may begin generating complete filter outputs as soon as the needed partial filtered results 210-240 are available. The CPU 302 may communicate with the partial filter result generator 304 to determine when sufficient number of partial filtered results 210-240 have been generated. When enough partial filtered results 210-240 have been generated, the CPU 302 may instruct the filter output generator 306 to begin generating the complete filter outputs. When any of the partial filtered results 210-240 are not needed for further generation of complete filter outputs, then they may be deleted from the memory 308 to save memory space.

[0026] The filter output generator 306 may process the partial filtered results 210-240 in the fast and slow scan directions as shown in Figs. 6 and 7. Thus, 210 (0 0), 220 (0 1), 230 (1 0) and 240 (1 1) are first accessed to generate the top left most complete filtered output pixel (0 0). The filter output generator 306 cannot begin generating this pixel until 240 (1 1) is generated by the partial filter result generator 304. However, after 240 (1 1) is generated, the filter output generator 306 may generate one complete output pixel for each

N/2 x N/2 block of image data processed by the partial filter result generator 304. After the complete filter output (0 0) is generated, the partial filtered results 210-240 for the (0 0) N/2 x N/2 block of image data may be deleted, because these partial filtered results are not needed to generated any other complete filter outputs. Similarly, the partial filtered results 210-24- (0 1) may be deleted after the completed filter output (0 1) is generated, and so on.

[0027] The complete filter outputs may either be stored in the memory 308 or output through the input/output port 310 to following processes for further processing. In this case, only a relatively small amount of memory is required if the filter device 300 filters the image data in a "on-the-fly" manner. N/2 x N/2 blocks of image data effectively stream into the filter device 300 as complete filter outputs stream out of the filter device 300 with un-needed N/2 x N/2 blocks of image data and partial filtered results overwritten by new N/2 x N/2 blocks of image data and newly generated partial filtered results.

[0028] As noted above, the filter processes discussed above may all be implemented by a software program executing in a processor such as the CPU 302. The software program may be stored on a computer readable medium or may be in a carrier wave form encoded to perform the described functions. All the functions of the partial filtered result generator 304 and the filter output generator 306 may be performed by software routines and the memory management of the N/2 x N/2 blocks of image data, partial filtered results 210-240 and the quad coefficients 110-140 may be easily performed by software. For example, if the sub-sampling ratio is k^d:1 where d is the dimension of the FIR filter, then a memory manager may delete k^d image data pixels that was partial filtered by partial filters (quad filters for d equals to 2) to generate the partial filtered results. Of course this memory management function may also be performed by corresponding hardware devices. The software program may be loaded from a computer readable medium such as a magnetic disk such as a floppy or an optical disk such as a CD, or transmitted electronically via a carrier wave on mediums such as the Internet, for example.

[0029] Figs 9-11 illustrates exemplary flow charts of the processes that may be executed by a hardware filter device or by a software program. Fig. 9 shows an exemplary process for preparing the quad coefficients. In step S100, zero padding is performed is N is odd. As discussed above, one row and one column of zeros may be attached to the top and left sides or bottom and right sides of the N x N block of coefficients to convert the N x N

block of coefficients to an N+1 x N+1 block of coefficients. The process goes to step S102. In step S102, the quad coefficients 110-140 are generated and stored in memory 308, for example. Then the process goes to step S104 and ends.

[0030] Fig. 10 shows an exemplary flow chart for generating partial filtered results. In step S200, a next N/2 x N/2 block of image data is received, and the process goes to step S202. In step 202, the partial filtered results are generated by multiplying each of the quad coefficients with corresponding image data pixels in the N/2 x N/2 block. The products corresponding to each of the quad coefficients are summed, and the process goes to step S204. In step S204, the process determines whether more N/2 x N/2 blocks of image data are to be processed. If more are to be processed, then the process returns to step S200; otherwise, the process goes to step S206 and ends.

[0031] Fig. 11 shows and exemplary process for generating complete filter outputs. In step S300, the process generates address mapping for accessing appropriate partial filtered results corresponding to each of the complete filter output pixels. This address mapping may be easily imbedded into a software program in terms of loops and increments. Corresponding hardware structures in the form of counters and gates may also be used. If the addressing of appropriate partial filtered results is either embedded in the software or built into hardware, this step may not be necessary. The process goes to step S302.

[0032] In step 302, the process reads appropriate one of the partial filtered results and sum the read partial filtered results. The sum is normalized by dividing by a sum of the coefficient values. This division may be performed by right shifting by an exponent of a power of 2 if the sum of the coefficients is the power of 2. If not a power of 2, then a fraction having a power of 2 denominator closest to the sum of the coefficients may be used. The normalized sum may be rounded by adding a value equal to half of the sum of the coefficients before the division process by either right shifting or multiplying by a fraction. The normalized and rounded sum is output as a complete filter output pixel. This normalization and rounding process may be performed by a software program or by a hardware unit (normalizer and a rounding device which may be an ASIC or digital logic, for example) Then the process goes to step S306. In step S306, the process determines whether all the desired complete filter outputs have been generated. If all the desired

complete filter outputs have been generated, then the process goes to step S308 and ends; otherwise the process returns to step S302.

[0033] The above described filter process may be used in applications such as xerographic marking devices or digital photocopiers. For example, de-screening of documents from half tone frequencies may be used in these and other applications. In these applications, FIR filters may be used to blur the image data to remove the half tone effects and/or to generate control signals by performing various filtering operations to obtain contrast information, for example. When so applied, an output of a peak and valley detector of the image data may be filtered by a FIR filter and the complete filter outputs may be multiplied by a DotGain parameter to convert the complete filter outputs to frequency units, for example. The application of quad filters may permit speed efficiencies by reducing repeated memory accesses to retrieve the same image data multiple times. Memory requirements may also be reduced a smaller amount of the image data is required to be maintained in memory. The partial filtered results being of much smaller volume.

[0034] While the invention has been described in conjunction with exemplary embodiments, these embodiments should be viewed as illustrative, not limiting. For example, while the above discussion used a two dimensional FIR filter as an example, any FIR filters of any number of dimensions may take advantage of the disclosed benefits. For example, a one dimensional FIR filter having N coefficients may be divided into two half partial filters having N/2 coefficients each. A three dimensional FIR filter may be divided into 6 sixth partial filters. In addition, various modifications, substitutes or the like are possible within the spirit and scope of the invention.